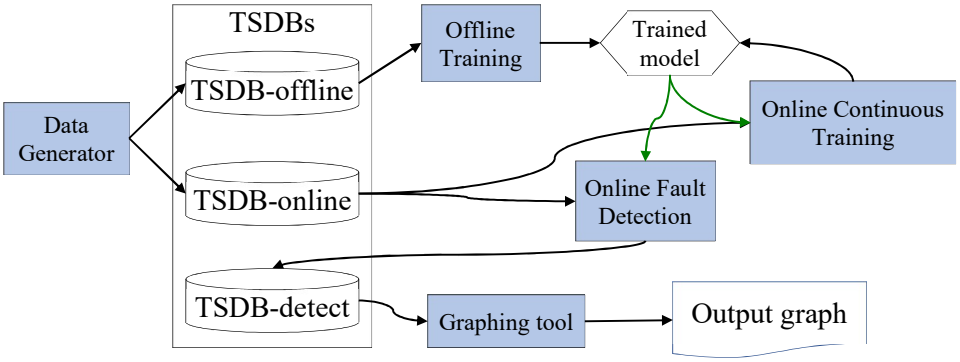
IoT Course Project 2

High level sketch

- Project concept
  - IoT is widely used in industry for problem analysis and fault diagnosis
  - This project simulates an online fault detection procedure under continuous input
  - Sensors are placed around to get input readings
    - Could be from machines, the environment, or detecting activities
    - Sensors could be hardware sensors or software sensors
    - E.g., hardware sensors: reading the pressure, temperature, etc. in a combustion engine
    - E.g., software sensors: detecting what activities are performed in a system
  - Classifiers are trained to recognize normal and failure conditions from sensor inputs
  - Sensor data may have trend change
    - For example, environment change may impact sensor readings
    - For example, a system getting worn out may have different normal sensor readings compared to when the system is new
    - So, for example, for failure detection, we cannot just use the pre-collected data for training, but also need continuous training from newly collected data
- Offline training for failure detection
  - Assume that we already have some labeled data for recognizing normal and failure cases
  - Feed the data with the labels to a neural network to learn to recognize failures
  - Let the labeled data be D1 and the trained model be M1
- Online failure detection and continuous training
  - Data continuously got collected from the sensors (let D2 be the new data), we need to
    - (1) Apply the existing model obtained earlier (M1) to D2 to determine whether the system is healthy or has failures
    - (2) Assume that some expert will determine whether the failure detection results are correct and correct them if they are wrong. To improve the trained model and to support proper learning while data trend shifts, we need to feed newly labeled D2 to the neural network and get new model M2
    - (3) Replace M1 by M2 and apply it to newly collected data D3 in the next time round
  - Perform steps (1)-(3) iteratively
- Data storage
  - New data gets collected continuously, and they should be fed to a storage for upcoming analytics and potential further processing
    - Kafka is a frequently used storage for continuous data processing
    - Some use other NoSQL databases for the storage, but it is less natural
    - Some uses the TSDB for the storage
    - For this project, we will use InfluxDB (a TSDB) for continuous time series data storage
- Example data source and data processing for this project
  - http://data-acoustics.com/measurements/bearing-faults/bearing-5/
    - The site includes normal datasets and faculty datasets, note that the normal or faulty conditions cannot be recognized by individual data points, but are recognized by the pattern in the data during a certain time window
  - Paper: Gear Fault Diagnosis Using Discrete Wavelet Transform and Deep Neural Networks



Project

- Input data
  - Data generator
    - A data generator will be offered, it generates a continuous data stream
      - TA will provide the data generator
    - The data stream is generated to simulate the sensor data coming into the system continuously
    - The stream mixes normal and faulty bearing data
      - Let DW be the data window size that is sufficient for special data patterns
    - The data generator randomly selects the normal and abnormal data from the files and merge them into a continuous data stream
      - Let FW be the fetch window size
      - The data generator fetches FW data as a unit from each file for merging
      - FW is a multiple of DW
    - The data generator also takes failure rate f as a parameter
      - In the generated data, f of the data are from faulty files and $(1-f)$ of them are from normal data files
  - Use the data generator to generate offline training dataset and add them into the TSDB
  - Use the data generator to generate the continuous data stream
    - When the data buffer is full, add them to the TSDB
    - Let BW be the buffer window size, only for writes
    - BW can be the same as DW or DW is a multiple of BW or BW is a multiple of DW, depending on the sensor data collection frequency and database write performance concern
  - Window sizes should be parameters to the system for exploration
- Software tools to be used for this project
  - Neural network PyTorch, https://pytorch.org/tutorials/
  - TSDB: InfluxDB, https://docs.influxdata.com/influxdb/v1.7/
- Offline Training
  - Read the data from TSDB-offline and feed them to the learning system
    - Use the method in the paper to perform failure detection training
      - Gear Fault Diagnosis Using Discrete Wavelet Transform and Deep Neural Networks
    - You can choose to skip the wavelet transform, which is not the focus of this project, but it would be nice if you do include that part
    - You will need to divide the data to feed to the neural network
      - The data window size should be DW
      - DW should be large enough so that the normal/faulty pattern can be recognized
  - Obtain the trained model M1 and pass it to the Online unit
- Online Fault Detection
  - Read the data from TSDB-online periodically (with size DW) and process the data

        - Need to setup a sync mechanism so that whenever the TSDB receives DW data, the processing element is activated to read the data of size DW
        - Explore what the TSDB has offered for easy sync, for example, the continuous query can be used for this purpose (or any other method)
        - Let the current dataset to be processed be Dcur and the current trained model be Mcur
        - Apply Mcur to Dcur to determine whether there is a fault and if so, the fault type
        - Write the fault detection results to the TSDB
- Online training
    - Fetch the current data with window size RTW (re-training window size)
        - RTW should be a multiple of DW (let RTW = k * DW)
        - Fetching RTW should be done whenever the data becomes available, using the same sync mechanism offered in the TSDB to do this
    - Add label to the data (the original fault label, not the label from your detection result)
        - The label is actually already available, but you do not use it during the detection phase
    - Feed the labeled data to the neural network to perform continuous training and modify Mcur into the next model
    - Pass the new model to the online fault detection element
- Online fault detection result presentation
    - Use the graphing utilities provided by the TSDB to graph each measurement
        - You can also use external tools
    - Fault data graphing (accurate labels)
        - Your online data, after being labeled, can be fed back to TSDB
        - Set normal = 0, different fault types as different positive integers
    - Fault data graphing (fault detection results)
        - Graphing the data obtained from your online fault detection (in TSDB)
        - Use the same fault value settings as above
        - Layout the two fault data series next to each other and in different colors to allow visual comparisons
        - Best is to have online graphing
    - Fault diagnosis result reporting
        - Output the fault diagnosis analysis accuracy in various metrics in a text window
    - You can explore different parameters and report these results